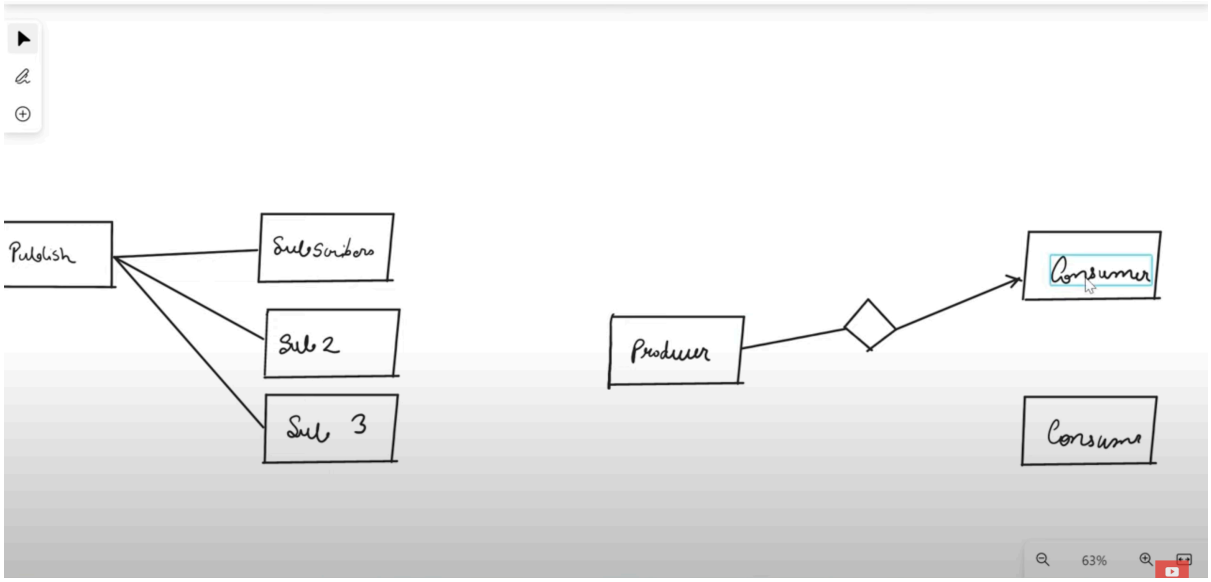
# What is Kafka

- Apache Kafka is publish-subscribe based fault tolerant messaging system. It is fast, scalable and distributed by design.
- It was initially thought of as a message queue and open-sourced by LinkedIn in 2011. Its community evolved Kafka to provide key capabilities:
  - Publish and Subscribe to streams of records, like a message queue.
  - Storage system so messages can be consumed asynchronously. Kafka writes data to a scalable disk structure and replicates for fault-tolerance. Producers can wait for write acknowledgments.
  - Stream processing with Kafka Streams API, enables complex aggregations or joins of input streams onto an output stream of processed data.

Traditional messaging models are queue and publish-subscribe. In a queue, each record goes to one consumer. In publish-subscribe, the record is received by all consumers.



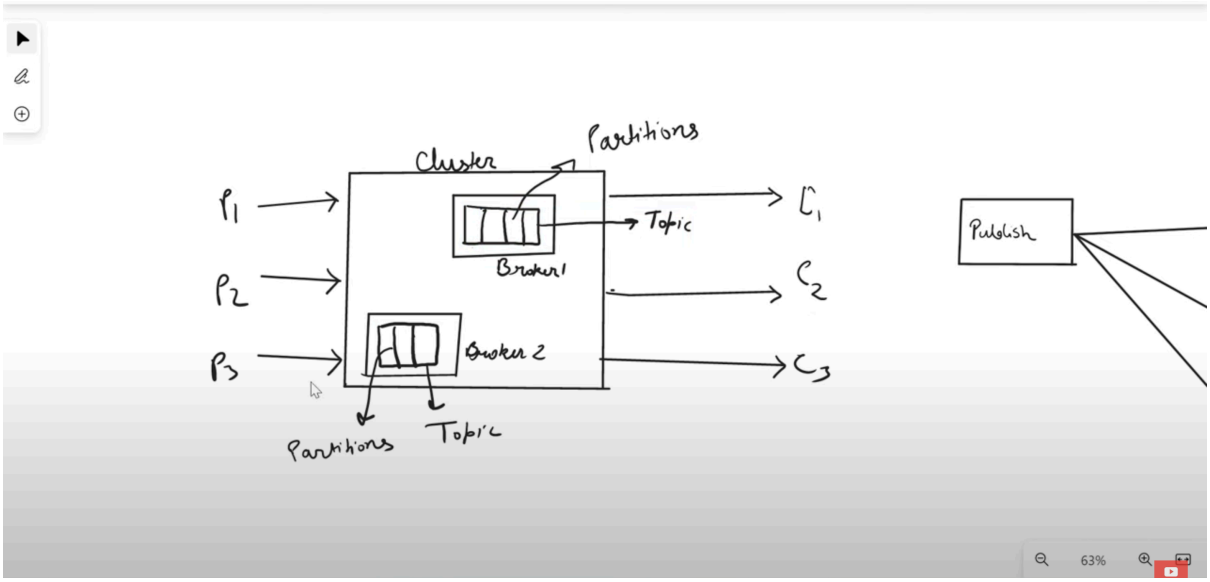
# Pros of Kafka

- Loose coupling — Neither service knows about each other regarding data update matters.
- Durability — Guarantees that the message will be delivered even if the consumer service is down. Whenever the consumer gets up again, all messages will be there.
- Scalability — Since the messages get stored in a bucket, there is no need to wait for responses. We create asynchronous communication between all services.
- Flexibility — The sender of a message has no idea who is going to consume it. Meaning you can easily add new consumers (new functionality) with less work.



# Cons of Kafka

- Semantics — The developer needs to have a deep understanding of the message flow as its strict requirements. Complex fallback approaches may take place.
- Message Visibility — You must track all those messages to allow you to debug whenever a problem occurs. Correlation IDs may be an option.



# Active MQ Vs RABBIT MQ VS Kafka communication

| Features | Active MQ/Rabbit MQ | Kafka |
|---|---|---|
| Developed By | Active mq - Created by founders from LogicBlaze, as an open-source message broker, later donated to Apache Software Foundation, where founders continue to develop the code base<br>Rabbit MQ- RabbitMQ is now owned by Pivotal Software Inc | Initially developed by LinkedIn. Later in In 2011 Kafka was open sourced and developed via the Apache Software Foundation hence named Apache Kafka. |
| Written in | Active mq - Java<br>Rabbit mq - Erlang | Java and Scala |
| Type PTP/ Pub-sub Messaging System | PTP Messaging System | Publish - Subscriber. Apache Kafka allows publishing and subscribing to streams of records. |



# Active MQ Vs RABBIT MQ VS Kafka communication

| Features | Active MQ/Rabbit MQ | Kafka |
|---|---|---|
| Overview | It is a traditional messaging system that deals with a small amount of data.<br><br>With point to point messaging, one or more consumers are connected to the queue, while the broker uses the 'round robin' approach to direct messages to specific consumers. | Kafka is a distributed publish-subscribe message delivery and logging system that follows a publisher/subscriber model with message persistence capability. Producers push event streams to the brokers, and consumers pull the data from brokers.<br><br>In subscription-based messaging, brokers broadcast messages to all consumers 'subscribed' to the topic. |
| Recommendation | Use them for exactly once delivery and for valuing messages | Use Kafka for high-performance monitoring and where losing messages is not important |





# Active MQ Vs RABBIT MQ VS Kafka communication

| Features | Active MQ/Rabbit MQ | Kafka |
|---|---|---|
| Throughput | Lower throughput since each delivery message state is maintained. | Higher throughput since producers dont wait for acknowledgements from brokers. So brokers can write messages at a very high rate. |
| Order of Messages | Cannot assure the order of messages to remain the same | Can assure that the messages retain the order in which they are sent |
| Filtering | Work of filtering is done by the JMS API message selector and not by the applications | No fundamental filters at the brokers. Hence it must be done by consumers or apps |
| Messaging Type | Push type messaging platform wherein the providers push the messages to consumers | Pull type messaging platform wherein consumers pull the message from the brokers |

# Active MQ Vs RABBIT MQ VS Kafka communication

| Features | Active MQ/Rabbit MQ | Kafka |
|---|---|---|
| Responsible Stakeholders | Responsibility of producers to ensure delivery of messages | Responsibility of consumers to consume messages |
| Scalability | No chances of horizontal scalability and replication | Scalable and available because of replication of partitions |
| Performance | Performance slows down as the number of consumers starts increasing. 4K-10K messages per second | Performance remains effective and good even if there are newer consumers being added. 1 million messages per second |
| Support for synchronous or asynchronous | Supports both | Supports asynchronous |

adsense keyword pl... 10 Features in Java... Nagarro Yashi outlo... 5 Hidden Secrets in... Some Java 8 Strea... Interview Preparati... HRIS Oracle Applica... Timeshhet Yammer »

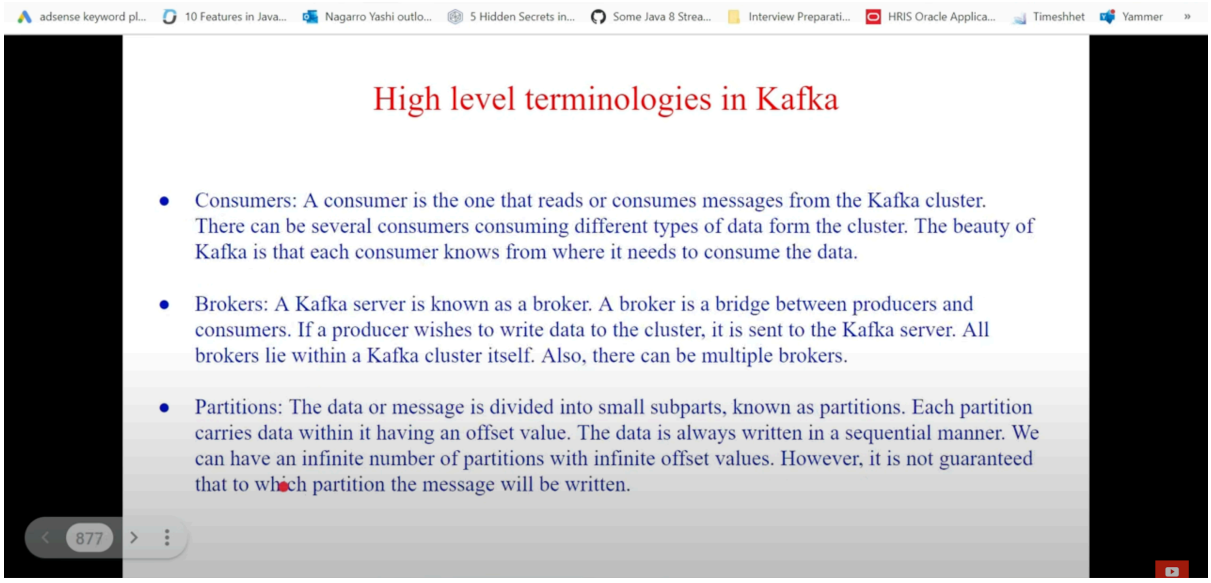
# Active MQ Vs RABBIT MQ VS Kafka communication

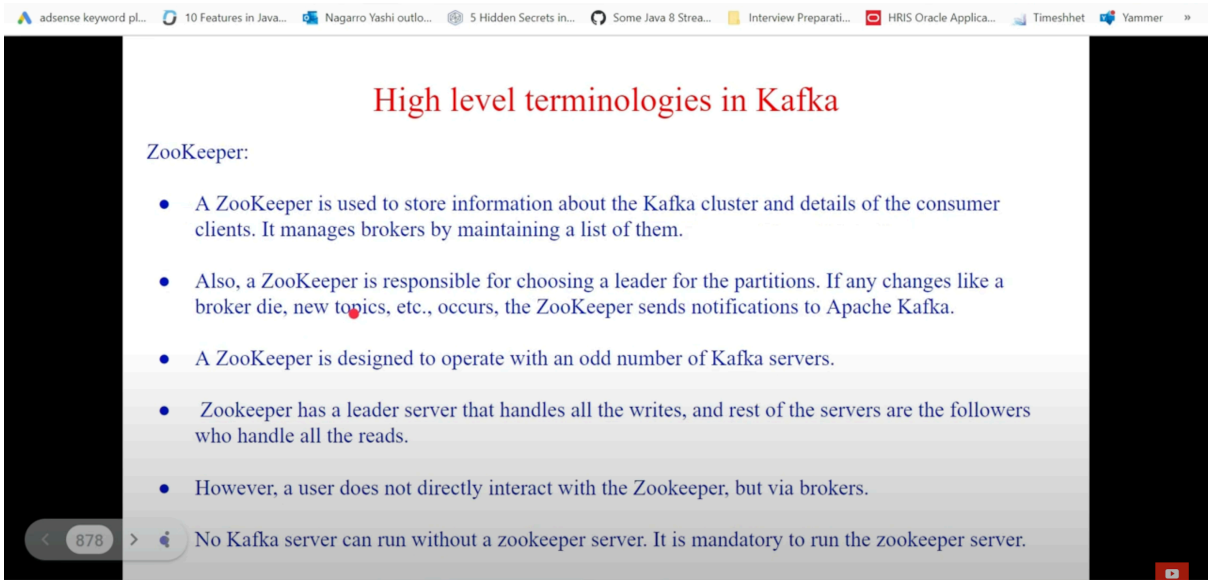
| Features | Active MQ/Rabbit MQ | Kafka |
|---|---|---|
| Message Retention | Acknowledgment based. These messages are removed from the queue once they are processed and acknowledged. | Policy-based (e.g retain only for 30 days). Kafka messages are always forever saved until the retention time expires, the developers have no other option. Kafka is a log. |
| Consumer Mode | Smart broker/dumb consumer. The broker consistently delivers messages to consumers and keeps track of their status | Dumb broker/smart consumer. Kafka doesn't monitor the messages each subscriber has read. Rather, it retains unread messages only, preserving all messages for a set amount of time. |
| Payload Size | No constraints | Default 1MB limit |
| Usage Cases | Simple use cases | Massive data/high throughput cases |

adsense keyword pl... 10 Features in Java... Nagarro Yashi outlo... 5 Hidden Secrets in... Some Java 8 Strea... Interview Preparati... HRIS Oracle Applica... Timeshhet Yammer »

# High level terminologies in Kafka

- Topic — A bucket of messages (or events in case of kafka) where services may place or read messages from. Messages are organized and durably stored in topics. A topic is similar to a folder, and these messages are the files in that folder. Topics are multi-producer and multi-subscriber, meaning that we can have zero, one or many of them both. Messages can be read as many times as needed, unlike traditional messaging systems, msgs / events in kafka are not deleted after consumption. Instead, you can define for how long Kafka should retain those events/msgs.
- Kafka Cluster: A Kafka cluster is a system that comprises of different brokers, topics, and their respective partitions.
- Producers: A producer sends or writes data/messages to the topic within the cluster. In order to store a huge amount of data, different producers within an application send data to the Kafka cluster.


adsense keyword pl...
10 Features in Java...
Nagarro Yashi outlo...
5 Hidden Secrets in...
Some Java 8 Strea...
Interview Preparati...
HRIS Oracle Applica...
Timeshhet
Yammer
High level terminologies in Kafka
Consumers: A consumer is the one that reads or consumes messages from the Kafka cluster. There can be several consumers consuming different types of data form the cluster. The beauty of Kafka is that each consumer knows from where it needs to consume the data.
Brokers: A Kafka server is known as a broker. A broker is a bridge between producers and consumers. If a producer wishes to write data to the cluster, it is sent to the Kafka server. All brokers lie within a Kafka cluster itself. Also, there can be multiple brokers.
Partitions: The data or message is divided into small subparts, known as partitions. Each partition carries data within it having an offset value. The data is always written in a sequential manner. We can have an infinite number of partitions with infinite offset values. However, it is not guaranteed that to which partition the message will be written.
877


adsense keyword pl...
10 Features in Java...
Nagarro Yashi outlo...
5 Hidden Secrets in...
Some Java 8 Strea...
Interview Preparati...
HRIS Oracle Applica...
Timeshhet
Yammer
High level terminologies in Kafka
ZooKeeper:
A ZooKeeper is used to store information about the Kafka cluster and details of the consumer clients. It manages brokers by maintaining a list of them.
Also, a ZooKeeper is responsible for choosing a leader for the partitions. If any changes like a broker die, new topics, etc., occurs, the ZooKeeper sends notifications to Apache Kafka.
A ZooKeeper is designed to operate with an odd number of Kafka servers.
Zookeeper has a leader server that handles all the writes, and rest of the servers are the followers who handle all the reads.
However, a user does not directly interact with the Zookeeper, but via brokers.
No Kafka server can run without a zookeeper server. It is mandatory to run the zookeeper server.
878


adsense keyword pl...
10 Features in Java...
Nagarro Yashi outlo...
5 Hidden Secrets in...
Some Java 8 Strea...
Interview Preparati...
HRIS Oracle Applica...
Timeshhet
Yammer
Architecture of Kafka
Kafka has a sized component called a cluster.
Inside a cluster, we've got several servers, also known as brokers (usually at least three brokers to provide enough redundancy.)
when a message gets sent to a broker, it gets sent to a particular topic.Topics allow us to categorize data. The data can eventually get broken into several partitions.
By splitting the topic into multiple partitions, we're facilitating scalability, as each partition can be read by a separate consumer.
Each broker is responsible for receiving messages from producers and committing those messages to disk. The broker is also responsible for answering consumers' fetch requests and serving them.
879